Лабораторна робота №7

ІПЗ-21-5 Богайчук Денис

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи неконтрольованої класифікації даних у машинному навчанні.

Завдання 1. Кластеризація даних за допомогою методу k-середніх

Лістинг коду програми

```
        import numpy as np
import matplotlib

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics

# Завантаження вхідних даних
X = np.loadtxt('data_clustering.txt', delimiter=',')

num_clusters = 5

# Включення вхідних даних до графіка
plt.figure()
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='black', s=80)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title('Вхідні дані')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

# Створення об'єкту KMeans
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)

# Навчання моделі кластеризації KMeans
kmeans.fit(X)

# Визначення кроку сітки
step_size = 0.01

# Відображення точок сітки
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_vals, y_vals = np.meshgrid(np.arange(x_min, x_max, step_size), np.arange(y_min,
y_max, step_size))

# Передбачення вихідних міток для всіх точок сітки
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

# Графічне відображення областей та виділення їх кольором
output = output.reshape(x_vals.shape)
plt.figure()
plt.clf()
```

```
plt.imshow(output, interpolation='nearest',
           extent=(x_vals.min(), x_vals.max(),
                   y_vals.min(), y_vals.max()),
           cmap=plt.cm.Paired,
           aspect='auto',
           origin='lower')

# Відображення вхідних точок
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='black', s=80)

# Відображення центрів кластерів
cluster_centers = kmeans.cluster_centers_
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], marker='o', s=210,
linewidths=4, color='black', zorder=12,
            facecolors='black')
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title('Границя кластерів')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

Результат виконання коду програми

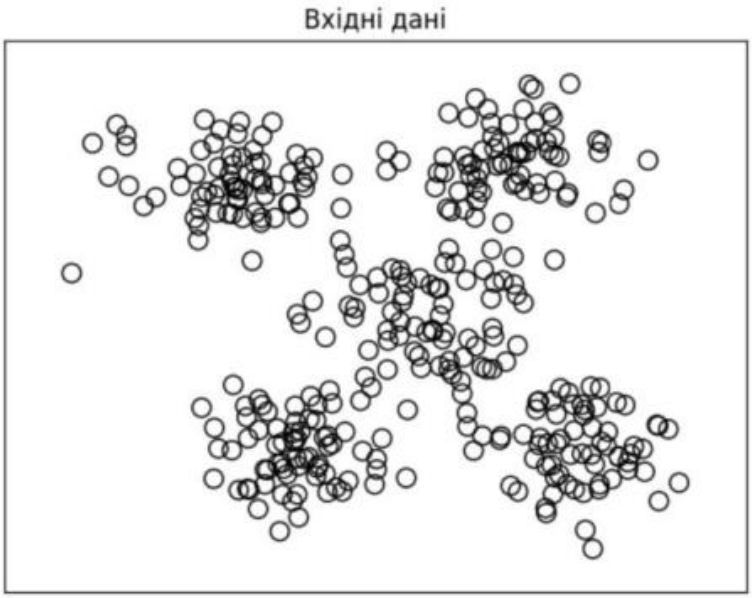


Рис. 1. Результат виконання програми

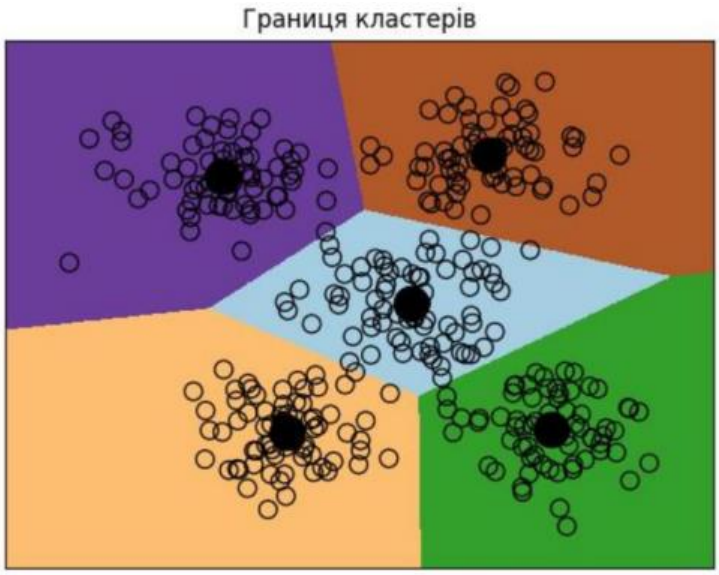


Рис. 2. Результат виконання програми

Результати виконання програмного коду виявилися доволі вдалими: більшість точок чітко розташовані в межах заданої області, а визначені центроїди точно відображають області з найбільшою щільністю точок у відповідних кластерах.

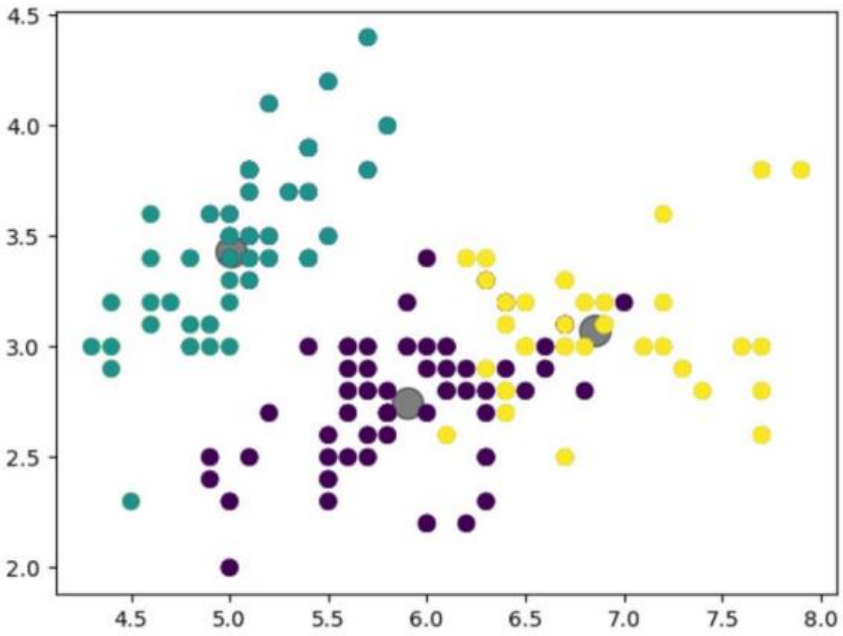
Завдання 2. Кластеризація K-середніх для набору даних Iris

Лістинг коду програми

```
        from sklearn.svm import SVC
from sklearn.metrics import pairwise_distances_argmin
import numpy as np
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt

iris = load_iris()
X = iris['data']
y = iris['target']

# Створення об'єкту KMeans
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10)

# Навчання моделі кластеризації KMeans
kmeans.fit(X)

# Передбачення вихідних міток
y_kmeans = kmeans.predict(X)

# Відображення вхідних точок
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
```

```
# Відображення центрів кластерів
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)


def find_clusters(X, n_clusters, rseed=2):
    # Довільне обрання кластерів
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # Призначення міток на основі найближчого центру
        labels = pairwise_distances_argmin(X, centers)

        # Знаходження нових центрів за середніми точками
        new_centers = np.array([X[labels == i].mean(0)
                                for i in range(n_clusters)])

        # Перевірка на збіжність
        if np.all(centers == new_centers):
            break
        centers = new_centers

    return centers, labels


# Відображення точок
centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
labels = KMeans(3, random state=0).fit predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
```

Результат виконання коду програми



Результати виконання програмного коду виявилися посередніми, але визначені центроїди коректно відображають області з максимальною щільністю точок у кожному кластері.

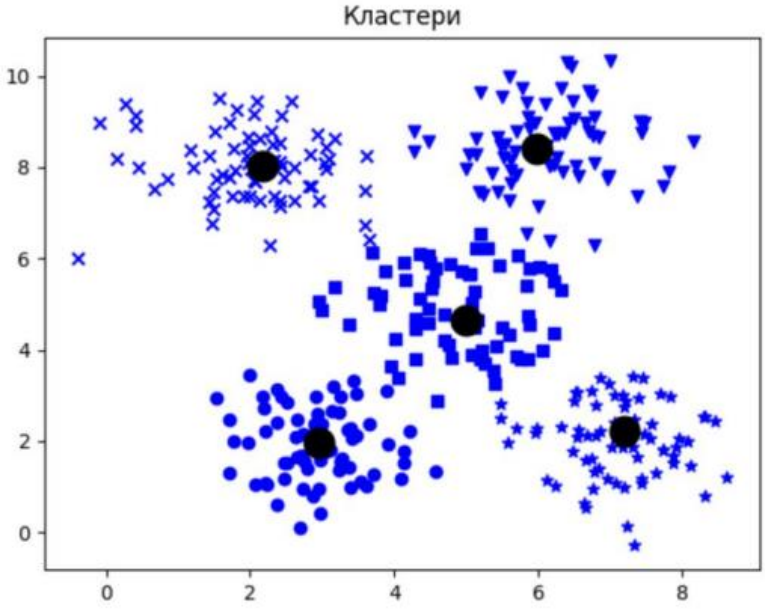
Завдання 3. Оцінка кількості кластерів з використанням методу зсуву середнього

Лістинг коду програми

```
        import numpy as np
import matplotlib

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle

# Завантаження даних
X = np.loadtxt('data_clustering.txt', delimiter=',')

# Оцінка ширини вікна для X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

# Кластеризація даних методом зсуву середнього
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)

# Витягування центрів кластерів
cluster_centers = meanshift_model.cluster_centers_
print('\nCenter of clusters:\n', cluster_centers)

# Оцінка кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
print('\nCenter of clusters in input data =', num_clusters)

# Відображення на графіку точок та центрів кластерів
plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), markers):
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color='blue')
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o',
markerfacecolor='black', markeredgecolor='black',
             markersize=15)
plt.title('Кластери')
plt.show()
```

Результат виконання коду програми

```
Center of clusters:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]

Center of clusters in input data = 5
```



Результати демонструють ефективність методу кластеризації на основі зсуву середнього. Успішно було ідентифіковано 5 кластерів, що відповідає кількості, заданій вручну в попередніх завданнях.

Завдання 4. Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

Лістинг коду програми

```
        import datetime
import json
import numpy as np
from sklearn import covariance, cluster
import matplotlib

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt
from matplotlib.finance import quotes_historical_yahoo_ochl as quotes_yahoo

# Вхідний файл із символічними позначеннями компаній
input_file = 'company_symbol_mapping.json'

# Завантаження прив'язок символів компаній до їх повних назв
with open(input_file, 'r') as f:
    company_symbols_map = json.loads(f.read())

symbols, names = np.array(list(company_symbols_map.items())).T

# Завантаження архівних даних котирувань
start_date = datetime.datetime(2003, 7, 3)
end_date = datetime.datetime(2007, 5, 4)
quotes = [quotes_yahoo(symbol, start_date, end_date, asobject=True) for symbol in
symbols]

# Вилучення котирувань, що відповідають відкриттю та закриттю біржі
opening_quotes = np.array([quote.open for quote in quotes]).astype(np.float)
closing_quotes = np.array([quote.close for quote in quotes]).astype(np.float)

# Обчислення різниці між двома видами котирувань
quotes_diff = closing_quotes - opening_quotes
X = quotes_diff.copy().T
X /= X.std(axis=0)

# Створення моделі графа
edge_model = covariance.GraphicalLassoCV()

# Навчання моделі
with np.errstate(invalid='ignore'):
    edge_model.fit(X)

# Створення моделі кластеризації на основі поширення подібності
_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()

for i in range(num_labels + 1):
    print("Cluster", i + 1, "==>", ','.join(names[labels == i]))
```

Висновок: У ході виконання лабораторної роботи було досліджено методи неконтрольованої класифікації даних у машинному навчанні з використанням Python та спеціалізованих бібліотек. Аналіз підтвердив ефективність методів кластеризації, таких як зсув середнього, у виявленні структур у даних.